DOI: 10.13656/j.cnki.gxkx.1997.04.016
广西科学 Guangxi Sciences 1997, 4 (4): 303~305

ORACLE SQL* FORMS触发器运行分析
Operation Analysis of ORACLE SQL* FORMS Trigger

刘智斌
Liu Zhibin

(广西大学计算机与信息工程学院 南宁市西乡塘东路 10号 530004)
(College of Computer and Information Engineering, Guangxi University, 10 East Xixiangtanglu, Nanning, Guangxi, 530004; E-mail: zbliu@gxu.edu.cn)

摘要 对 ORACLE SQL* FORMS触发器运行的分析,表明触发器的运行机制与普通的程序设计技术既有相似之处,又有区别。正确掌握触发器的运行机制是触发器设计过程的重要一环。

关键词 ORACLE SQL* FORMS 触发器 表单 块 域

Abstract The principles and features of trigger were discussed by analyzing the operation of ORACLE SQL* FORMS Trigger. They share similarity with and are also different from the design technique of ordinary program. It is necessary to make sure the operation of triger in its design.

Key words ORACLE, SQL* FORMS, trigger, form, block, field
中图法分类号 TP 311.13

1 SQL* FORMS触发器简介

触发器(Trigger)是由用户编写的用于增加应用功能的代码块。触发器顾名思义,其中的代码块必须在某一条件具备的情况下才触发执行,因此每一种触发器必须与一个事件发生联系, ORACLE SQL* FORMS能识别一组预先定义好的运行事件(如: 光标离开某个域,或光标进入某个域等),每个事件又对应一个内部触发器,每个触发器都有自己的名字,并由一条或多条 PL/SQL语句和 SQL* FORMS命令组成。当用户为应用编写触发器时,必须先确定激活该触发器的事件,因为触发器的名字就是由激活它的事件来决定的。

每一个触发器还必须连接到表单(FORM)中某个具体的对象上,这些对象包括一个域或字段(Field),一个块(Block),或 FORM本身,所连接的对象决定了触发器的作用范围。

触发器的功能既像过程又有别于过程。过程需通过显式的调用才被执行,而触发器只要在触发它的事件具备时便立刻被触发,同时系统自动执行写在触发器中的命令。另外触发器必须依附于某个模块对象而存在,而过程则不需要。触发器是新一代数据库开发工具常备的一种新的设计工具,它借鉴了一些面向对象的设计技术。

1997-08-06收稿。

2 触发器的定义级别和作用范围

ORACLE SQL* FORMS中为触发器提供了三种定义级别,由低级到高级依次是: 域级,块级和 FORM级。

触发器的定义级别决定了它的作用范围。当同一事件定义了不同级别的触发器时,具有最低级别的触发器优先执行。例如: 当在某一个域上定义了一个域级(field level) Post-Field触发器,那么只有当光标离开此域时,该触发器就被触发。

3 触发器运行分析

由于触发器中的代码是在触发它的事件发生时自动被执行,因此了解与掌握正确的触发时刻在触发器设计中是至关重要的。以下通过一些实例来说明这一问题。首先让我们来看一个在 ORACLE SQL* FORMS 3.0环境下做的实验:

创建一个 FORM取名为 TEST,并为 TEST建一个块,取名为 B, B块不基于任何基表,并且由

NO、NAME和AGE三个域(字段)组成,NO为第一个域,NAME为第二个域,AGE为第三个域。现在分别在NO和NAME域上各设计一个触发器。在NO域上设计一个名字为Key-nxtfld的触发器。该触发器在按Enter或Tab键时被激活,也就是按下回车键或Tab键将光标移到下一个域时被激活。该触发器的内容如下:

```
Field name: NO; Trigger: Key-nxtfld
: Global.A = '1'; /* 给全局变量A赋初值为字符'1' */
Go_field('NAME'); /* 将光标移出该域转到名为NAME的域中 */
: Global.A = '3'; /* 刷新全局变量A使其等于字符'3' */
```

在NAME域设计的触发器名字为On-new-field-instance该触发器在光标移到某个新的域时被激活。该触发器的内容如下:

```
Field Name: NAME; Trigger: On-new-field-instance
Message(: Global.A); /* 在屏幕下方显示全局变量A的当前内容 */
```

设计完毕后,先将TEST FORM存盘并随后生成它(Generate)。然后便可在DEBUG MODE方式下运行TEST FORM。此时会发现屏幕下方显示的内容是字符'3'而非字符'1'。一般认为,当执行到Go____field('NAME')命令时,光标应该立刻离开NO域而移到NAME域上并激活On-new-field-instance触发器,此时屏幕显示的应是字符'1'而非字符'3'。但显示结果并非这样,为什么?在回答这一问题之前,我们先看另一个实验:

即把NAME域上的On-new-field-instance触发器改名为Pre-Field触发器,而触发器的内容不变,仍为Message(: Global.A),既:

```
Field Name: NAME; Trigger: Pre-Field
Message(: Global.A); /* 在屏幕下方显示全局变量A的当前内容 */
```

修改完毕后,重新运行TEST。此时,屏幕下方显示的内容不再是字符'3';而是字符'1'。这一结果与前面设想的执行流程相符合。这说明当在NAME域上定义的是On-new-field-instance触发器时,NO域上的Key-nxtfld触发器执行到Go_field('NAME')命令时,并不能将执行流程转到NAME域上而激活在NAME域上定义的On-new-field-instance触发器,而是继续执行Go_field('NAME')的后续语句,待Key-nxtfld触发器的内容执行完毕后,才将光标移到NAME域上继而触发On-new-field-instance触发器并执行该触发器中的命令,因此显示的结果是字符'3'。但当在NAME域上定义的是Pre-Field触发器,执行流程就能像所设想的那样,因Go_field('NAME')命令而立刻转到NAME域上的Pre-Field触发器并执行Message(: Global.A),因此显示的结果为字符'1'。

以上结果表明,NO域的Key-nxtfld触发器中的Go_field('NAME')命令不能立刻激活On-new-field-instance触发器,却能立刻激活NAME域上的Pre-Field触发器。原因就在于这两种触发器的激活条件不同。On-new-field-instance触发器是当光标已到达某个新域后才触发,而Pre-Field触发器是当光标还没到达但即将到达某域时就可被触发。另外实验还表明在ORACLE的SQL* FORMS中,光标的移动滞后于触发器中命令的执行。因此,当执行Go_field('NAME')命令后,由于光标滞后于命令的执行,而不能立刻移到NAME域上而激活On-new-field-instance触发器,但激活NAME域的Pre-Field触发器的条件已满足,因此Pre-Field触发器被立刻激活并抢在流程还没执行后继命令之前将其转去执行Pre-Field触发器中的命令,而On-new-field-instance触发器要等到NO域的Key-nxtfld触发器的所有命令执行完毕后才移动光标继而被触发。

为了更进一步地了解触发器的运行机制,下面再看一个实验,首先在NO、NAME和AGE三个域上设计如下几个触发器:

```
Field name: NO; Trigger: Key-nxtfld
: Global.A = '1'; /* 给全局变量A赋初值为字符'1' */
Go_field('NAME'); /* 将光标转到NAME域上 */
Message('no1 '||: Global.A); /* 在屏幕下方显示全局变量A的当前内容 */
: Global.A = '3'; /* 刷新全局变量A使其等于字符'3' */
Go_field('AGE'); /* 将光标转到AGE域上 */
Message('no2 '||: Global.A); /* 在屏幕下方显示全局变量A的当前内容 */
: Global.A = '4'; /* 刷新全局变量A使其等于字符'4' */
Field Name: NAME; Trigger: Pre-field (以下省略注解)
```

```
    Message ('no3 ||: Global. A);
  : Global. A: = 2;
Field Name: NAME; Trigger: On- new- field- in-
stance
    Message ('no4 ||: Global. A);
Field Name: AGE; Trigger: Pre- field
    Message ('no5 ||: Global. A);
  : Global. A: = 5;
```

然后运行 TEST FORM,此时光标停在 NO域上,按 Enter键后,屏幕下端依次显示以下结果:

no3 1no1: 2
no5 3
no2 5
no4 4

最后光标停在 NAME域上。

首先让我们分析一下运行结果:当光标在 NO域上按 Enter键时,便立刻触发 NO上的 Key- nxtfld触发器,并执行该触发器中的命令,当执行到 Go_ field ('NAME')时,触发 NAME域的 Pre- field触发器并显示 no3 1,且将: Global. A改为 2,然后执行流程自动返回到 NO域的 Key- nxtfld触发器并继续执行第三条命令即显示 no1: 2;最后按照相同的原理继续执行 Key- nxtfld触发器中的后续语句并依次显示 no5 3 no2 5和 no4 4

如果在 NO域上增加一个 Post- field触发器,其内容如下:

```
Field name: NO; Trigger: Post- field
    Message ('no6' ||: Global. A);
```

重新运行 TEST FORM发现:首先显示的结果是 no6 1接着才是 no3 1,这说明当执行到 Go_ field ('NAME')命令时,首先触发 NO域的 Post- field触发器,然后才触发 NAME域的 Pre- field触发器,而这两个触发器的触发时刻非常接近。

如果将 AGE域上的 Pre- field触发器的内容改为:

```
Field Name: AGE; Trigger: Pre- field
    Message ('no5 ||: Global. A);
  : Global. A: = 5;
    Go_ field ('NO');
  : Global. A: = 6;
```

并添加一个新的触发器如下:

```
Field Name: AGE; Trigger: On- new- field- in-
stance
    Message ('no7: ||: Global. A);
```

再次运行 TEST FORM,将发现屏幕下端显示的结果依次为:

no6 1
no3 1
no1: 2
no5 3
no3 5
no2 2
no4 4

对这一运行结果首先要作解释的是为什么会显示两次 no3 . . . ,即 NAME域的 Pre- field触发器被触发两次。这是因为在 Pre或 Post等导航类触发器中不能使用导航类命令所引起的。导航类命令包括: Go_ block Go_ field Next_ block和 Next_ field等命令,这类命令的执行会引起导航事件的发生并因此激活导航类触发器。如果导航类触发器中又含有导航类命令,则这类命令势必要产生新的导航事件,而这将与原来的导航事件发生冲突。因此每当出现这种情况,ORACLE系统将会报错,并使导致这一现象的第一个导航类命令的导航功能无效,同时使导航状态返回到上一次的导航状态中。结合以上实验,当流程执行到 AGE的 Pre- field触发器中的 Go_ field ('NO')命令发生导航冲突时,系统将立刻报错,终止流程继续执行并重新返回到 NO域的 Key- nxtfld触发器中由 Go_ field ('NAME')所产生的导航状态中,因而第二次触发 NAME域的 Pre- field触发器,显示 no3 5后,再沿着 Go_ field ('AGE')的后继语句继续执行。由于 Go_ field ('AGE')的导航功能已无效,因而光标最后只能按照 Go_ field ('NAME')的导航功能移到 NAME字段上最后激活该域的 On- new- field- instance触发器并显示 no4 4 以上运行结果进一步表明了触发器的运行与普通过程既有相似之处又有明显的区别。

4 结束语

触发器是新一代的数据库开发工具常备的一种工具。它与普通的程序设计技术既有相似只处,又有区别。因此正确分析和掌握触发器的运行机制是触发器设计过程中至关重要的一环。

参考文献

1 ORACLE 7. 0版数据库及其开发工具系统资料.
2 孙宏昌等. ORACLE应用系统开发工具. 北京: 清华大学出版社, 1995.

(责任编辑: 蒋汉明)